

Chapter 27

Darshan

Philip Carns

Argonne National Laboratory

27.1	Features	311
27.2	Success Stories	313
27.3	Conclusion	315
	Bibliography	316

Darshan is an application-level I/O characterization tool that captures production-level I/O behavior with minimal overhead. Darshan does not record a complete trace of all I/O system calls. It instead gathers compact access pattern statistics for each file opened by the application. These statistics are reduced, compressed, and aggregated into a single log file that summarizes the I/O activity and access patterns of the application as a whole. Although this summary data does not offer the same fidelity as a traditional tracing or profiling tool, it can be collected with negligible overhead and no source code modification. This combination of features makes it possible not only to instrument full-scale application runs, but also to transparently deploy Darshan for the automatic characterization of all production jobs on a leadership-class HPC system. Darshan characterization data can be used for a variety of purposes ranging from performance tuning of specific applications [6, 7, 8] to analysis of trends in system-wide I/O behavior [1, 2].

Although Darshan is designed for system-wide deployment, it can also be installed and used by individual end-users as well. The runtime component of Darshan consists of a set of user-space libraries and compiler wrappers to simplify development environment integration. The command line utility component of Darshan includes tools to interpret application logs and produce graphical summaries for high-level analysis.

27.1 Features

The initial motivation for the Darshan project was to gain a better understanding of production I/O behavior by performing system-wide workload studies. Previous system-wide workload studies [9] were very influential in HPC I/O research but no longer reflected the scale, architecture, and scientific application diversity of present-day systems. Collecting data on large-scale present-day systems required the development of efficient, non-intrusive instrumentation methods. This led directly to the following core design goals for Darshan: transparent integration with the user environment and negligible impact on application performance, and reliability.

Darshan operates in the user-space as an interposition library in order to collect per-application statistics without source code modifications. As with many other HPC profiling tools, Darshan leverages the MPI profiling interface in conjunction with either link-time wrappers for statically linked executables or preloaded libraries for dynamically linked executables. Static instrumentation can be enabled system-wide using MPI compiler script functionality, while dynamic instrumentation can be enabled system wide using environment variables. End-users do not need to change their work flow in either case.

The Darshan function call wrappers intercept POSIX and MPI-IO functions, as well as a few key HDF5 and PNetCDF functions. The wrappers are used to gather information, such as operation counters (such as **open**, **read**, **write**, **stat**, and **mmap**); datatypes and hint usage; access patterns in terms of alignment; sequentiality; access size; and timing information including cumulative I/O time and intervals of I/O activity. (A full description of counters can be found in the Darshan documentation [5].) Darshan does not issue any communication or storage operations to manage characterization data while the application is running. Each process is instrumented independently using a bounded amount of memory. When the application shuts down, the results are then aggregated, compressed, and stored persistently. Darshan uses a combination of MPI reduction operations, collective I/O, and parallel Zlib compression to reduce overhead and minimize log file size.

The command line utility component of Darshan includes tools to parse and analyze log files produced by the runtime library. Figure 27.1 shows an example of output produced by **darshan-job-summary**, a utility that summarizes the I/O behavior of a job. This example was chosen from production logs captured on the Mira IBM Blue Gene/Q system operated by the Argonne Leadership Computing Facility (ALCF). The “I/O Operation Counts” graph in the upper right corner indicates that the MPI-IO collective buffering optimization [10] was enabled; there is a large discrepancy between the number of MPI-IO collective write calls and the number of POSIX write calls. The “Most Common Access Sizes” table confirms that the majority of the POSIX write operations were 16 MB in size, which corresponds to the collective buffer

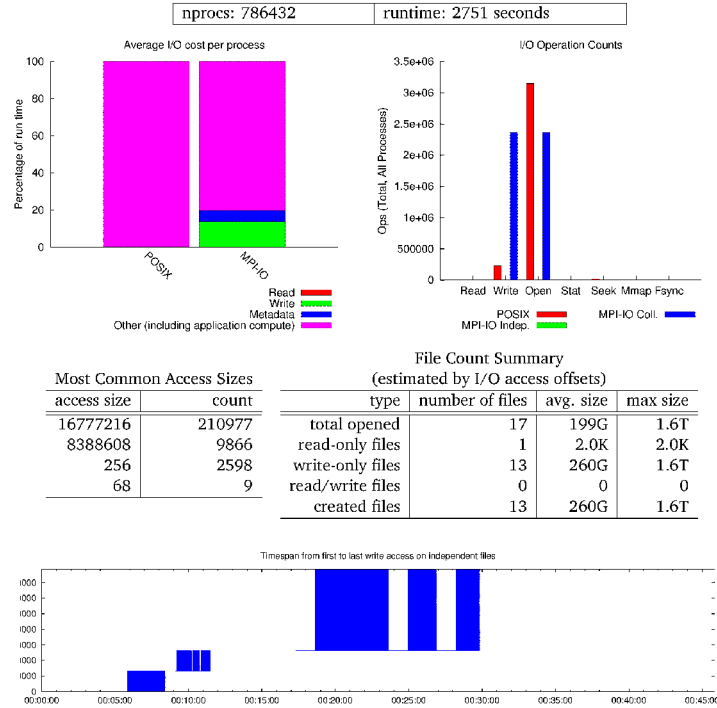


FIGURE 27.1: Screenshot showing excerpts from the `darshan-job-summary` utility included with Darshan. This example summarizes the I/O behavior of a 786,432 process turbulence simulation that wrote over 3 TB of data.

size used by MPI-IO. The bottom graph indicates that the application was divided into subsets of processes that wrote data in different time intervals.

The Darshan command line utilities also include tools to anonymize identifying information, such as file names and executable names, within log files. This capability makes it possible to release Darshan characterization data to the community. The ALCF I/O Data Repository [3] is a notable example. It provides public access to nearly 200,000 log files collected on the Intrepid Blue Gene/P system at the ALCF.

27.2 Success Stories

Darshan can fundamentally change the approach to I/O performance engineering on systems where it is deployed. When an I/O performance problem is observed, scientists and I/O experts can immediately refer to the Darshan

report for initial diagnosis. This often eliminates the need for additional profiling runs (costly in CPU time) or manual source code inspection (costly in manpower).

Historic collections of Darshan characterization data can also be used to study trends in system usage as well, as illustrated in this section with example data collected on the ALCF Intrepid Blue Gene/P system. Intrepid is a 557-teraflop system containing 163,840 compute cores, 80 TB of RAM, and 7.6 PB of storage. More information about Intrepid and its I/O subsystem can be found in Chapter 4. Darshan has been used to automatically instrument MPI applications on Intrepid since January 2010. In this case study, five months of Darshan log files are analyzed from January 1, 2013 to May 30, 2013. Darshan instrumented 13,613 production jobs from 117 unique users over this time period.

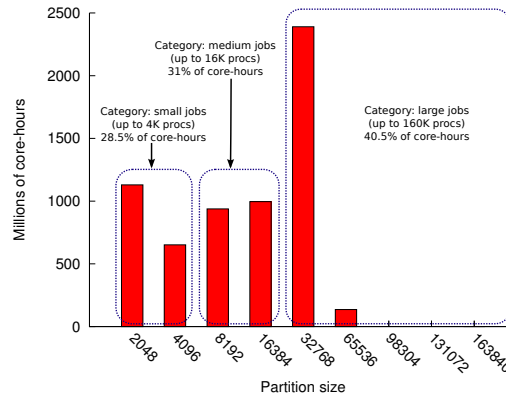


FIGURE 27.2: Histogram of the number of core-hours consumed by Darshan-instrumented jobs in each partition size on Intrepid. The histogram bins are further categorized into small, medium, and large sizes for subsequent analysis.

The jobs instrumented by Darshan in the first five months of 2013 ranged in size from 1 process to 163,840 processes, offering an opportunity to observe how I/O behavior varies across application scales. Figure 27.2 shows a histogram of the number of core-hours consumed by Darshan-instrumented jobs in each of the 9 available partition sizes on Intrepid. The most popular partition size in terms of core-hour consumption contains 32,768 cores. The jobs can be split into three comparably sized, broader categories for further analysis; however, 28.5% of all core hours were in partitions of size 4,096 cores or smaller, 31% of all core hours were in partitions of size 8,192 or 16,384, and 40.5% of all core hours were in partitions of size 32,768 or larger.

Figure 27.3 shows the total amount of data read and written by jobs in each partition size category. All three categories are dominated by write activity with one notable exception: the small job size category is dominated by a single climate application labeled as “Climate user A.” This application accounted

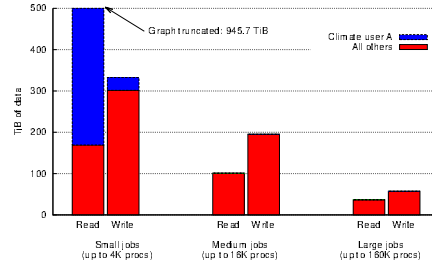


FIGURE 27.3: Total amount of data read and written by Darshan-instrumented jobs in each partition size category on Intrepid.

for a total of 776.5 TB of read activity and 31.1 TB of write over the course of the study by reading as much as 4.8 TB of NetCDF data in each job instance. The other notable trend evident in Figure 27.3 is that smaller jobs accounted for a larger fraction of the I/O usage on the system than larger jobs.

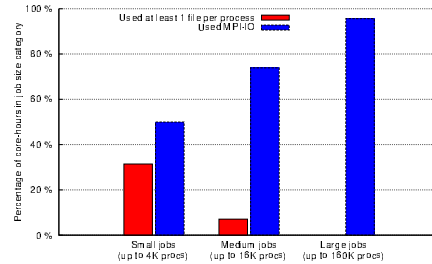


FIGURE 27.4: Prevalence of key I/O characteristics in each partition size category on Intrepid.

Figure 27.4 illustrates the prevalence of two key I/O characteristics across job size categories. The first is file-per-process file usage. A job was defined as having a file-per-process access pattern if it opened at least N files during execution, where N is the number of MPI processes. Such jobs account for 31% of all core hours in the small job size category, but they do not appear at all in the large job size category. Another job was defined as using MPI-IO if it opened at least one file using `MPI_File_open()`. In contrast to the file-per-process usage pattern, MPI-IO usage increases with job scale, going from 50% for small jobs up to 96% for large jobs. The decline in file-per-process access patterns and the increase in MPI-IO usage suggest that large-scale applications are using more advanced I/O strategies in order to scale effectively and simplify data management.

27.3 Conclusion

The Darshan I/O characterization tool has demonstrated that it is possible to instrument leadership-class production applications with negligible overhead. Since its initial development in 2009 [4], it has been in production on multiple large-scale HPC systems, including the IBM Blue Gene/P systems, IBM Blue Gene/Q systems, and Cray XE6 systems. Darshan enables both targeted investigation of key applications, as well as broad system studies. Key challenges for the future are to further expand the scope of such instrumentation without compromising efficiency and to develop more sophisticated tools to leverage data produced by Darshan.

Bibliography

- [1] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross. Understanding and Improving Computational Science Storage Access Through Continuous Characterization. *ACM Transactions on Storage (TOS)*, 7(3):8, 2011.
- [2] P. Carns, Y. Yao, K. Harms, R. Latham, R. Ross, and K. Antypas. Production I/O Characterization on the Cray XE6. In *Proceedings of the Cray User Group meeting 2013 (CUG 2013)*, 2013.
- [3] Philip Carns. ALCF I/O Data Repository. Technical report, Argonne National Laboratory (ANL), 2013.
- [4] Philip Carns, Robert Latham, Robert Ross, Kamil Iskra, Samuel Lang, and Katherine Riley. 24/7 Characterization of Petascale I/O Workloads. In *Proceedings of 2009 Workshop on Interfaces and Architectures for Scientific Data Storage*, September 2009.
- [5] Darshan Documentation. <http://www.mcs.anl.gov/research/projects/darshan/documentation/>, 2013.
- [6] Jing Fu, Misun Min, R. Latham, and C.D. Carothers. Parallel I/O Performance for Application-Level Checkpointing on the Blue Gene/P System. In *The workshop on Interfaces and Abstractions for Scientific Data Storage (IASDS) in conjunction with the 2011 IEEE International Conference on Cluster Computing*, pages 465–473, 2011.
- [7] Rob Latham, Chris Daley, Wei-keng Liao, Kui Gao, Rob Ross, Anshu Dubey, and Alok Choudhary. A Case Study for Scientific I/O: Improving

the FLASH Astrophysics Code. *Computational Science & Discovery*, 5(1):015001, 2012.

- [8] Ning Liu, Jing Fu, Christopher D. Carothers, Onkar Sahni, Kenneth E. Jansen, and Mark S. Shephard. Massively Parallel I/O for Partitioned Solver Systems. *Parallel Processing Letters*, 20(04):377–395, 2010.
- [9] Nils Nieuwejaar, David Kotz, Apratim Purakayastha, Carla Schlatter Ellis, and Michael Best. File-Access Characteristics of Parallel Scientific Workloads. *IEEE Transactions on Parallel and Distributed Systems*, 7(10):1075–1089, October 1996.
- [10] Rajeev Thakur, William Gropp, and Ewing Lusk. Data Sieving and Collective I/O in ROMIO. In *The Seventh Symposium on the Frontiers of Massively Parallel Computation*, pages 182–189. IEEE, 1999.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. Please make sure you have the DOE acknowledgment at the end of your paper (before the References).

For acknowledgment information, please use:

This work was supported by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357.